THE HECKMAN BINDERY, INC.
North Manchester, Indiana

KRL

H or V	JUST	FONT	SLOT	TITLE
H	CC	1W	22	BEBR
			21	FACULTY
			20	WORKING
			19	PAPER
H	CC	1W	8	1990
			7	NO.1705-1718
H	CC	1W		330
				B385<"CV">
				no.1705-1718
				cop.2
				<IMPRINT>
H	CC	7W		U. of ILL.
				LIBRARY
				URBANA

BINDING COPY

PERIODICAL ☐ CUSTOM ☐ STANDARD ☐ ECONOMY ☐ THESIS | NO VOLS THIS TITLE | LEAD ATTACH

BOOK ☐ CUSTOM ☐ MUSIC ☐ ECONOMY | AUTH. 1ST ☐

ACCOUNT	LIBRARY	NEW	RUB OR SAMPLE	TITLE I.D	COLOR FOIL	COLOR MATERIAL
66672 001				6632	WHI	488

ACCOUNT NAME
UNIV OF ILLINOIS

ACCOUNT INTERNAL I.D. | ISSN.

B01912400

I.D #2 | NOTES | BINDING FREQUENCY | WHEEL | SYS. I.D.

STX3 | 1 | 3 | 39256

COLLATING
35

ADDITIONAL INSTRUCTIONS
Dept=STX3 Lot=#20 Item=151 HNM=|BY#
1CR2ST3CR MARK BY # B4 91

SEP. SHEETS	PTS. BD. PAPER	TAPE STUBS	CLOTH EXT	FILLER GUM	FILLER STUB

POCKETS: PAPER | BUCK | CLOTH
SPECIAL PREP
LEAF ATTACH

INSERT MAT
ACCOUNT LOT NO: #20
JOB NO: HV363
PRODUCT TYPE: 11
ACCOUNT PIECE NO: 151
PIECE NO: 81
HEIGHT: 11.2
GROUP CARD: 1
VOL THIS TITLE:
COVER SIZE: V X 9

00124752

FACULTY WORKING PAPER 90-1714

Determination of Optimal Due Dates and Sequence

Suresh Chand
Purdue University

Dilip Chhajed
University of Illinois
Department of Business Administration

Bureau of Economic and Business Research
College of Commerce and Business Administration
University of Illinois at Urbana-Champaign

BEBR

FACULTY WORKING PAPER NO. 90-1714

College of Commerce and Business Administration

University of Illinois at Urbana-Champaign

December 1990

Determination of Optimal Due Dates and Sequence

Suresh Chand
Purdue University

and

Dilip Chhajed
University of Illinois at Urbana-Champaign
Department of Business Administration

This research was supported in part by the Center for the Management of Manufacturing Enterprises at Purdue University. Several insightful comments by Professor J.G. Shanthikumar were helpful in revising the paper.

ABSTRACT

The problem of simultaneous determination of optimal due dates and optimal sequence for N-job single-machine problem with multiple due dates is considered in this paper. The penalty for a job is assumed to be a linear function of the due date and the earliness/tardiness for the job. The objective is to minimize the total penalty for all jobs. An efficient optimal algorithm to solve the problem is developed and several important results are proved.

INTRODUCTION

This paper considers the problem of simultaneous determination of optimal due dates and optimal sequence for N independent jobs to be processed on a single machine. All jobs are available at time zero. Processing times are known and deterministic. The penalty for a job in a sequence is assumed to be a linear function of the due date assigned to the job and the earliness/tardiness for the job in the sequence. The same linear penalty function is used for all jobs. The objective is to minimize the total penalty for all jobs. The number of distinct due dates, m, to be assigned to the jobs is assumed to be prespecified and known. We allow m to take any value in the set {1,2,...,N}. Job preemption is not allowed.

The problem of simultaneous determination of optimal due dates and optimal sequence has been motivated and considered by several authors in the recent literature [1, 2, 6, 9, 10]; the reader is directed to Baker and Scudder [3] and Bector, Gupta and Gupta [4] for a brief review of this literature. The problem considered in this paper is most directly related with the problems considered in Panwalkar, et al. [5], Seidman, et al. [7], and Bagchi [1]. Panwalkar, et al. [5] assume a common due date for all jobs; that is, m = 1. The penalty function used in this paper is the same as in [5]. Several results from [5] are used in this paper. As in [5], we show that the penalty function can be expressed as a product of two sequences, which is minimized by arranging them in opposite order. Seidman, et al. [7] consider the problem when each job is allowed a different due date. They show that an SPT (shortest processing time) sequence is optimal for this problem. With the number of distinct due dates, m, permitted to be any number in the set {1,2,...,N}, our paper, in essence, bridges the gap between the scheduling models with a common due date and the scheduling models with possibly N distinct due dates for the N-job problem.

Bagchi [1] assumes that the N jobs to be processed on the single machine are to meet the requirements of m customer orders. It is assumed that $m \leq N$ and that each customer order has one or more jobs in it. The specific jobs in each customer order are assumed to be known. A common due date is assumed for all jobs in a customer order. Thus, there could be m distinct due dates for

all jobs. Instead of the due date penalty used in our paper, Bagchi uses a lead time penalty which is a linear function of the lead time, where lead time for a job in a customer order is the completion time of the last job in the customer order. These assumptions allowed Bagchi to decompose the sequencing problem into two independent sequencing subproblems -- one to schedule the jobs within a customer order and the other one to schedule the customer orders. The first subproblem is shown to be the common due date problem of Panwalkar, et al. [5]. The second subproblem is shown to be the minimum weighted completion time problem which can be solved using the WSPT (weighted shortest processing time) algorithm.

Our paper considers two different cases. In the first case we assume that the decision maker specifies the number of jobs to be assigned to different due dates. However, unlike Bagchi [1], the specific jobs to be assigned to different due dates are unknown. In the second case, we assume that the number of jobs to be assigned to different due dates is unknown. It should be remarked that the second case turned out to be very difficult to analyze.

Section 2 of the paper defines the notation and formulates the problem.

Section 3 of the paper considers the case when the vector $(n_1,n_2,...,n_m)$, where n_j is the number of jobs assigned to the jth due date, is externally specified. Note that while n_j is assumed to be known, the specific jobs to be assigned to the jth due date are unknown and to be determined. It is shown that the objective function can be expressed as a sum of the pairwise product of two equal sets of numbers, where one set consists of processing times and the other set consists of positional weights. The positional weights are independent of job characteristics. An efficient procedure is presented to find optimal due dates and an optimal sequence. The results in this section are used in the next section in developing an optimal procedure when the vector $(n_1,n_2,\ldots,n_m)$ is also a decision variable.

Section 4 of the paper develops some dominance results to find an optimal vector $(n_1,n_2,...,n_m)$ when it is not externally specified. These dominance results are used in developing an optimal procedure to find $(n_1,n_2,...,n_m)$. Even though these results are intuitive, proofs for these turned out to be very difficult.

Section 5 of the paper discusses the computational requirements for the algorithm in Section 4. It is shown that our algorithm can give large computational savings compared to a complete search. In one example, we show that our algorithm is more than 10^8 times faster than a complete search.

The paper closes in Section 6 by providing some concluding remarks and ideas for further research.

2. PROBLEM FORMULATION

Throughout the paper, we assume that the number of jobs to be scheduled and the number of due dates to be assigned to the jobs are known. The following notation is used to formulate the problems:

N = Number of jobs to be scheduled,

m = Number of due dates,

$\langle n,n+k\rangle$ = $\{n,n+1,n+2,\ldots,n+k\}$,

d_i = Due date for job i, $i \in \langle 1,N\rangle$,

C_i = Completion time for job i, $i \in \langle 1,N\rangle$,

E_i = Max $(0,d_i-C_i)$ = Earliness for job i, $i \in \langle 1,N\rangle$,

T_i = Max $(0,C_i-d_i)$ = Tardiness for job i, $i \in \langle 1,N\rangle$,

t_i = Processing time for job i, $i \in \langle 1,N\rangle$,

P_1 = Per unit time due date penalty for each job,

P_2 = Per unit time earliness penalty for each job, and

P_3 = Per unit time tardiness penalty for each job.

In addition, we let $D_1 \le D_2 \le \ldots \le D_m$ denote the m due dates and let I_j denote the set of jobs assigned to due date D_j for $j \in \langle 1,m\rangle$. As in Panwalkar, Smith and Seidman [5], the constants P_1, P_2 and P_3 are assumed to be non-negative and known.

The objective is to determine $D = \{D_1,D_2,\ldots,D_m\}$, $I = \{I_1,I_2,\ldots,I_m\}$ and a schedule σ to minimize the total penalty $TP(D,I,\sigma)$, where

$$TP(D,I,\sigma) = \sum_{j=1}^{m} \sum_{i \in I_j} \left\{ P_1 \cdot D_j + P_2 \cdot E_i + P_3 \cdot T_i \right\} \quad (1)$$

and $d_i = D_j$ for $i \in I_j$.

It is easy to see that the problem has an optimal solution with zero machine-idle time. (See Lemma 2 in the Appendix.) Therefore, it is sufficient to consider permutation schedules to find optimal solutions.

Let [k] denote the job in position k in a permutation schedule, then

$$C_{[k]} = \sum_{i=1}^{k} t_{[i]}. \quad (2)$$

We will use σ to denote a permutation schedule in the rest of the paper. Also, jobs are assumed to be indexed such that

$$0 \le t_1 \le t_2 \le \ldots \le t_N \quad (3)$$

The next section considers the problem when $|I_j| = n_j$ is known for all $j \in \langle 1,m \rangle$. Note that while $|I_j|$ is assumed to be known, the specific jobs in I_j are unknown and to be determined.

3. RESULTS AND PROCEDURE WITH $|I_j| = n_j$ KNOWN

This section assumes that the vector $(n_1,n_2,\ldots,n_m)$ of m positive integers such that $\sum_{j=1}^{m} n_j = N$ and $|I_j| = n_j$ for $j = 1,2,\ldots,m$ is externally specified. D, I and σ are decision variables. The cases when m=1 and m=N are considered in Panwalkar, Smith and Seidman [5] and Seidman, Panwalkar and Smith [7], respectively.

For $P_1 \ge P_3$, an SPT sequence is optimal with $d_i = 0$ for $i \in \langle 1,N \rangle$ (see Theorems 1 and 2 in [7]). We assume $P_1 < P_3$ in developing the properties and the algorithm below.

Properties of Optimal Solutions: We now state several properties of optimal solutions which will be used in developing an optimal algorithm. Lemmas 1 and 3 needed to develop these properties are proved in the appendix. We define $N_j = \sum_{k=1}^{j} n_k$ for $j \in \langle 1,m \rangle$ with $N_0 = 0$; N_j is the total number of jobs assigned to the first j due dates.

From Lemma 1 in the Appendix, it is easy to see that for any given D and σ, there is an optimal I such that a batch of consecutive n_j jobs in σ is assigned to D_j. The following property gives an optimal I for any given D and σ.

Property 1: For any given D and σ, there is an optimal I such that

$$I_j = \left\{ \sigma_{N_{j-1}+1}, \sigma_{N_{j-1}+2}, \cdots, \sigma_{N_{j-1}+n_j} \right\}$$

for $j \in \langle 1,m \rangle$, where σ_n is the job in position n in sequence σ.

This property essentially says that there is an optimal solution such that n_j consecutive jobs (in positions $N_{j-1}+1$ to $N_{j-1}+n_j$) in σ are assigned to D_j.

Lemma 3 in the Appendix shows that for any given σ, there is an optimal D such that D_j coincides with the completion time of a job in σ. The following property, which gives an optimal D for a given σ, follows from Panwalkar, et al. [5].

Property 2: For a given σ, there is an optimal D such that $D_j = C_{[k_j]}$, where

$$k_j = N_{j-1} + \left(\frac{P_3-P_1}{P_3+P_2} \cdot n_j \right)^+ \qquad (4)$$

$(x)^+$ here stands for the smallest integer $\geq x$.

Property 2 implies that the first (k_j-N_{j-1}) jobs in I_j are early and the remaining (N_j-k_j) jobs are tardy.

We now develop an algorithm to find (D,I,σ) to minimize (1).

Algorithm: Using Properties 1 and 2, we first reduce (1) to the form $\sum_{i=1}^{N} w_i \cdot t_{[i]}$. Knowing the positional weights w_i's for positions $i \in \langle 1,N \rangle$, an optimal sequence can be found as in Panwalkar, Smith and Seidman [5].

Using Property 1, we get:

$$TP(D,\sigma) = \min_{I} TP(D,\sigma,I) = \sum_{j=1}^{m} \sum_{i=N_{j-1}+1}^{N_j} (P_1 \cdot D_j + P_2 \cdot E_{[i]} + P_3 \cdot T_{[i]}).$$

Further, using Property 2, we can write:

$$TP(\sigma) = \min_{D} TP(D,\sigma) = \sum_{j=1}^{m} \left[n_j \cdot P_1 \cdot C_{[k_j]} + \sum_{i=N_{j-1}+1}^{k_j} P_2 \cdot (C_{[k_j]} - C_{[i]}) + \sum_{i=k_j+1}^{N_j} P_3 \cdot (C_{[i]} - C_{[k_j]}) \right], \qquad (5)$$

with k_j defined by (4). Using $C_{[k]} = \sum_{i=1}^{k} t_{[i]}$ for a given σ, (5) can be simplified to:

$$TP(\sigma) = \sum_{i=1}^{N} w_i \, t_{[i]}, \tag{6}$$

where

$$w_i = \begin{cases} P_2 \cdot (i-1-N_{j-1}) + P_1 \cdot (N-N_{j-1}) & \text{for } i \in \langle N_{j-1}+1, k_j \rangle \\ P_3 \cdot (N_j - i + 1) + P_1 \cdot (N-N_j) & \text{for } i \in \langle k_j+1, N_j \rangle \end{cases} \tag{7}$$

Note that (6) is a product of two sequences, which is minimized by arranging them in opposite order. The following steps give an optimal solution for a given $(n_1,n_2,...,n_m)$. We call this algorithm the Given-$(n_1,n_2,...,n_m)$-Algorithm or the Gn-Algorithm.

Steps of the Gn-Algorithm

Step 0: Set $N_0 = 0$ and $N_j = \sum_{k=1}^{j} n_k$ for $j \in \langle 1,m \rangle$.

Step 1: For $j \in \langle 1,m \rangle$, compute

$$k_j = N_{j-1} + \left(\frac{P_3 - P_1}{P_3 + P_2} \cdot n_j \right)^{+}.$$

Step 2: For $i \in \langle 1,N \rangle$, compute w_i using (7).

Step 3: Rank the positional weights w_i's in a descending order of magnitude such that the largest w_i is ranked first and the smallest w_i is ranked N^{th}. Break the ties arbitrarily.

Step 4: Find a sequence σ by assigning job i to position k where k is such that w_k has rank i. Set

$D_j = \sum_{i=1}^{k_j} t_{[i]}$, and

$d_i = D_j$ for $i \in I_j = \left\{ \sigma_{N_{j-1}+1}, \sigma_{N_{j-1}+2}, \cdots, \sigma_{N_j} \right\}$

for $j \in \langle 1,m \rangle$. Stop, the solution found in Step 4 is an optimal solution.

It is easy to see that the first $(k_j - N_{j-1})$ jobs in I_j are in an LPT order while the last $(N_j - k_j)$ jobs are in an SPT order. We now give a numerical example to illustrate the procedure.

Numerical Example

Given 10 jobs with $t_1 = 3$, $t_2 = 10$, $t_3 = 11$, $t_4 = 13$, $t_5 = 13$, $t_6 = 16$, $t_7 = 17$, $t_8 = 20$, $t_9 = 22$ and $t_{10} = 25$. The penalties are $P_1 = 2$, $P_2 = 11$ and $P_3 = 18$. We are given $m = 2$, $n_1 = 4$ and $n_2 = 6$.

From Steps 0 and 1 of the algorithm, we get $N_1 = 4$, $N_2 = 10$, $k_1 = 3$ and $k_2 = 8$. The ten positional weights and their ranks are

Position	1	2	3	4	5	6	7	8	9	10
Weight w_i	20	31	42	30	12	23	34	45	36	18
Rank	8	5	2	6	10	7	4	1	3	9

Optimal Sequence: 8-5-2-6-10-7-4-1-3-9; $D_1 = t_8+t_5+t_2 = 43$, and $D_2 = t_8+t_5+t_2+t_6+t_{10}+t_7+t_4+t_1 = 117$; $I_1 = \{8,5,2,6\}$ and $I_2 = \{10,7,4,1,3,9\}$; Penalty = 3763.

Next section considers the problem when the vector $(n_1,n_2,...,n_m)$ is unknown.

4. RESULTS AND PROCEDURE WITH $|I_j| = n_j$ UNKNOWN

This section considers the problem when the vector $(n_1,n_2,...,n_m)$ is unknown. D, I, and σ are decision variables in this case. Dominance properties are developed to find optimal $n_1,n_2,...,n_m$. A key result in the section shows that there is an optimal solution with $n_1 \geq n_2 \geq ... \geq n_m$. This decreasing property is used to develop an efficient search algorithm to find optimal $n_1,n_2,...,n_m$. Computational requirements for this search algorithm are discussed.

Recall that w_i denotes the positional weight associated with position i for a known vector $(n_1,n_2,...,n_m)$. Step 3 of the Gn-Algorithm gives ranks for these weights. We need the following additional notation associated with these weights:

α_i = the positional weight with the rank of i

$$WT_i = \sum_{k=i}^{N} \alpha_k$$

Clearly $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_N$.

Note that the weights α_i's and WT_i's, while they depend on the vector $(n_1,n_2,...,n_m)$ and the penalties P_1, P_2 and P_3, they do not depend on the processing times for the jobs.

To develop the results in this section, we first assume m = 2. We let $WT_i(n)$ denote the weight WT_i when $n_1 = n$ and $n_2 = N-n$. $\alpha_i(n)$ is similarly defined. Let COST(n) denote the corresponding optimum objective function value, then we have

$$\begin{aligned} COST(n) &= t_1\cdot\alpha_1(n) + t_2\cdot\alpha_2(n) + \ldots + t_N\cdot\alpha_N(n) \\ &= t_1\cdot(WT_1(n)-WT_2(n)) + t_2(WT_2(n)-WT_3(n)) + \ldots + t_N\cdot(WT_N(n)) \\ &= t_1\cdot WT_1(n) + (t_2-t_1)\cdot WT_2(n) + (t_3-t_2)\cdot WT_3(n) + \ldots + (t_N-t_{N-1})\cdot WT_N(n) \\ &= t_1\cdot WT_1(n) + \sum_{i=2}^{N} \Delta i\cdot WT_i(n), \end{aligned} \tag{8}$$

where $\Delta i = t_i - t_{i-1}$.

Lemma 4: For m = 2, $n \in \langle 1,N-1\rangle$ and $\ell \in \langle 1,N-1\rangle$, we have $COST(n) \le COST(\ell)$ for every vector $(t_1,t_2,\ldots,t_N)$ satisfying (3) if and only if

$$WT_i(n) \le WT_i(\ell) \quad \text{for all } i \in \langle 1,N\rangle \tag{9}$$

Proof: From (8) we have

$$COST(\ell) - COST(n) = t_1\cdot(WT_1(\ell)-WT_1(n)) + \sum_{i=2}^{N} \Delta i\cdot(WT_i(\ell)-WT_i(n)) \tag{10}$$

If (9) holds for given n and ℓ, then $COST(\ell)-COST(n) \ge 0$ from (10) for any given $(t_1,t_2,\ldots,t_N)$ satisfying (3). If (9) does not hold for all $i \in \langle 1,N\rangle$, then there is a $k \in \langle 1,N\rangle$ such that $WT_k(n) > WT_k(\ell)$ or $WT_k(\ell)-WT_k(n) < 0$. Then, we can select a vector $(t_1,t_2,\ldots,t_N)$ satisfying (3) with $\Delta k = t_k - t_{k-1}$ high enough to make $COST(\ell)-COST(n) < 0$ in (10). This completes the proof.

Lemma 5: For m = 2 and $n \in \langle 0,\left(\frac{N}{2}\right)^+\rangle$, we have $WT_i(n) \ge WT_i\left(\left(\frac{N}{2}\right)^+\right)$ for all $i \in \langle 1,N\rangle$. (Recall that $(x)^+$ is the smallest integer $\ge x$.)

Proof: A proof for this is given in the appendix.

THEOREM 1: There is an optimal solution (D^*,I^*,σ^*) such that $n_1^* \ge n_2^* \ge \ldots \ge n_m^*$.

Proof: First consider the case m = 2. If $n_1 < \left(\frac{N}{2}\right)^+$, then $WT_i(n_1) \geq WT_i\left(\left(\frac{N}{2}\right)^+\right)$ for all $i \in \langle 1,N\rangle$ from Lemma 5, and $COST(n_1) \geq COST\left(\left(\frac{N}{2}\right)^+\right)$ from Lemma 4. Thus, there is an optimal solution with $n_1^* \geq \left(\frac{N}{2}\right)^+ \geq \frac{n_1^*+n_2^*}{2}$, or $n_1^* \geq n_2^*$.

The proof for m = 2 can be extended for the m > 2 case by considering two adjacent batches. Thus, there is an optimal solution with $n_j^* \geq n_{j+1}^*$ for j = 1,2,...,m-1, or $n_1^* \geq n_2^* \geq \ldots \geq n_m^*$.

The following search algorithm gives an optimal $(n_1^*,n_2^*,\ldots,n_m^*)$. We call this algorithm the Efficient Search Algorithm (or the ES-Algorithm).

Efficient Search Algorithm

For $n_1 = \left(\frac{N}{m}\right)^+, \left(\frac{N}{m}\right)^+ +1,\ldots,N-(m-1)$

For $n_2 = \left(\frac{N-n_1}{m-1}\right)^+, \left(\frac{N-n_1}{m-1}\right)^+ +1,\ldots,N-(m-2)$

For $n_k = \left(\frac{N-\sum_{j=1}^{k-1} n_j}{m-(k-1)}\right)^+, \left(\frac{N-\sum_{j=1}^{k-1} n_j}{m-(k-1)}\right)^+ + 1,\ldots,N-(m-k)$

For $n_m = N-\sum_{j=1}^{m-1} n_j$

Find minimum total penalty for $(n_1,n_2,\ldots,n_m)$ using the Gn-Algorithm

Find $(n_1^*,n_2^*,\ldots,n_m^*)$ corresponding to the lowest total penalty.

The following example illustrates the procedure.

Numerical Example: Consider the 10-job problem with processing-time and penalty data used in the numerical example in Section 3. We solve it for m = 2 and m = 3.

Solution for m = 2: To find an optimal (n_1^*, n_2^*), we need to find the minimum cost using the Gn-Algorithm for $n_1 = 5, 6, 7, 8$ and 9. Note that $n_2 = 10 - n_1$. The following table gives the costs using the Gn-Algorithm:

n_1	5	6	7	8	9
n_2	5	4	3	2	1
COST	3,624	3,586	3,729	4,121	4,738

We get $n_1^* = 6$ and $n_2^* = 4$ corresponding to the minimum cost of \$3,586.

Solution for m = 3: The costs computed using the Gn-Algorithm for various values of n_1, n_2 and n_3 in the ES-Algorithm are given below.

n_1	n_2	n_3	COST
4	3	3	2,757
4	4	2	2,852
5	3	2	2,845
5	4	1	3,023
6	2	2	3,021
6	3	1	3,068
7	2	1	3,367
8	1	1	3,829

We get $n_1^* = 4$, $n_2^* = 3$ and $n_3^* = 3$.

The next section discusses the computational requirements for the ES-Algorithm.

5. COMPUTATIONAL REQUIREMENTS FOR THE EFFICIENT SEARCH ALGORITHM

The ES-Algorithm generates vectors $(n_1, n_2, \ldots, n_m)$ by partitioning N into m integer numbers such that $\sum_{j=1}^{m} n_j = N$, and $n_1 \geq n_2 \geq \ldots \geq n_m \geq 1$. Let P_m^N denote the total number of these vectors, then the following theorem gives a recursive procedure to compute P_m^N.

THEOREM 2: For a positive integer $n \geq 2$ and $r \in \langle 1, n-1 \rangle$, we have

$$P_1^n = P_n^n = P_1^1 = 1, \text{ and}$$

$$P_r^n = P_1^{n-r} + P_2^{n-r} + \ldots + P_{\min(r,n-r)}^{n-r}. \qquad (11)$$

Proof: $P_1^n = 1$ because $n_1 = n$ is the only feasible partition. $P_n^n = 1$ because $n_1 = n_2 = \ldots = n_n = 1$ is the only feasible partition. We now prove (11).

We are given r due dates and $n \geq r$ jobs to be assigned to these due dates. Since each due date is assigned at least one job, we can first assign one job to each of the r due dates. We are left with n-r jobs.

P_1^{n-r} in the r.h.s. of (11) gives the total number of ways if the remaining n-r jobs are assigned to the first due date. P_2^{n-r} in the r.h.s. of (11) gives the total number of ways if the remaining n-r jobs are assigned to the first two due dates such that, out of n-r jobs, each due date gets at least one job and the first due date gets at least as many jobs as the second one. The remaining terms in the r.h.s. of (11) can be explained similarly.

Computational Savings: The ES-Algorithm requires P_m^N applications of the Gn-Algorithm to find $(n_1^*, n_2^*, \ldots, n_m^*)$. $\binom{N-1}{m-1}$ gives the total number of ways of assigning N jobs to m due dates such that each due date gets at least one job, so a complete enumeration to find $(n_1^*, n_2^*, \ldots, n_m^*)$ would require $\binom{N-1}{m-1}$ applications of the Gn-Algorithm. The ratio $\binom{N-1}{m-1}/P_m^N$ can be used as a measure of computational savings realized by the use of our dominance results in the ES-Algorithm. Table 1 gives the values of P_m^N, $\binom{N-1}{m-1}$ and $\binom{N-1}{m-1}/P_m^N$ for N = 40 and different values of m.

Results in Table 1 indicate that large computational savings can be realized by using our algorithm compared to a complete search. For N = 40 and m = 20, Table 1 indicates that the ES-Algorithm is more than 10^8 times faster than a complete search. (Shanthikumar and Yao [8] also noticed that the use of the decreasing property narrows down the search for an optimal solution to a very small subset of the solution space.)

Analysis for m = 1 and m = n:

For m = 1, we have $n_1^* = N$, and only one ($P_1^N = 1$) application of the Gn-Algorithm is required. It is easy to see that our algorithm reduces to the algorithm of Panwalkar, Smith and Seidmann [5] for the m = 1 case.

For m = N, we have $n_1^* = n_2^* = \ldots = n_N^* = 1$. For this case also, only one ($P_N^N = 1$) application of the Gn-Algorithm is required. It is easy to see that

$$N_j = n_1^* + n_2^* + \ldots + n_j^* = j \qquad \text{for } j = 1,2,\ldots,N,$$

$$k_j = j-1 + \left(\frac{P_3-P_1}{P_3+P_2}\right)^+ = j \qquad \text{for } j = 1,2,\ldots,N, \text{ and}$$

$$w_i = P_1 \cdot (N+1-i) \qquad \text{for } i = 1,2,\ldots,N.$$

TABLE 1: Table of Computational Savings for the ES-Algorithm for N = 40

m	P_m^N	$\binom{N}{m-1}$	$\binom{N}{m-1}/P_m^N$
1	1	1	1
2	20	39	1.95
3	133	741	5.18
4	478	9139	19.11
5	1115	82,251	73.76
6	1945	575,757	296.01
10	3590	2.12×10^8 (approx.)	5.90×10^4
15	1861	1.39×10^{10} (approx.)	7.46×10^6
20	627	6.36×10^{10} (approx.)	1.01×10^8

Since $w_1 > w_2 > \ldots > w_N$, the objective function is minimized by arranging the jobs in an increasing order of processing times. Thus, an SPT sequence is optimal for this case.

A Negative Result: Let COST(n) denote the minimum cost of the N-job, 2-due date problem with n jobs assigned to the first due date. In the numerical example in Section 4, COST(n) has the incline property; that is, as n increases, once COST(n) goes up, it does not come down. We found that COST(n) has this property for most of the problems that we solved.

Unfortunately, COST(n) does not always have this property as shown by the following counter-example. Further computational savings could be realized if COST(n) had the incline property.

Counter-Example: Assume N = 6; $P_1 = 17$, $P_2 = 11$, $P_3 = 18$; $t_1 = t_2 = t_3 = t_4 = t_5 = 5$, and $t_6 = 25$. The following table gives COST(n) for different values of n.

n	1	2	3	4	5
COST(n)	2,195	2,180	2,175	2,180	2,175

COST(n) in this example does not follow the incline property.

6. CONCLUDING REMARKS

This paper makes two major contributions. First, an optimal algorithm of the order O(N log N) was developed to determine optimal due dates and a sequence for an N-job problem when the number of jobs to be assigned to different due dates are known. It was shown that the objective function can be expressed as a sum of the pairwise product of two equal sets of numbers, where one set consists of processing times and the other set consists of positional weights.

Second, an important dominance property was developed to determine the number of jobs to be assigned to different due dates. While the result is intuitive, the proof turned out to be difficult. An algorithm of the order $O(P_m^N \cdot N \log N)$ was developed to find optimal due dates, sequence and number of jobs to be assigned to different due dates. Theorem 2 in Section 5 gives a computational procedure to find P_m^N. It was shown that large computational savings can be realized by our procedure compared to a complete search.

Appendix

This appendix states and proves several important results.

Lemma 1: Let D, σ and I be known. Let I be such that there are three successive jobs, a, b, and c ($C_a < C_b < C_c$) in σ with a, c $\in I_j$ and b $\notin I_j$. Then it is possible to find an alternate I (without changing D and σ) with the same or a lower cost such that either a, b $\in I_j$ and c $\notin I_j$ or b, c $\in I_j$ and a $\notin I_j$.

Proof. Let b $\in I_k$, k $\neq$ j. We provide a proof by considering the following 4 cases. C_a, C_b and C_c are the completion times for jobs a, b and c, respectively. D_j is the due date for jobs in I_j and D_k is the due date for jobs in I_k. We have $C_a < C_b < C_c$.

Case 1: Job b early in I_k and early if assigned to I_j, or $C_b \leq \min(D_j, D_k)$

$$\begin{aligned}\Delta(b) &= \text{COST of b in } I_k - \text{COST of b in } I_j \\ &= \left[P_1 \cdot D_k + P_2 \cdot (D_k - C_b)\right] - \left[P_1 \cdot D_j + P_2 \cdot (D_j - C_b)\right] \\ &= P_1 \cdot (D_k - D_j) + P_2 \cdot (D_k - D_j)\end{aligned}$$

In this case, since $C_a < C_b$, Job a is early in I_j and will be early if assigned to I_k.

$$\begin{aligned}\Delta(a) &= \text{COST of a in } I_k - \text{COST of a in } I_j \\ &= P_1 \cdot (D_k - D_j) + P_2 \cdot (D_k - D_j) \\ &= \Delta(b)\end{aligned}$$

Assigning Job a to I_k and Job b to I_j gives the same cost.

Case 2: Job b tardy in I_k and tardy if assigned to I_j, or $C_b \geq \max(D_k, D_j)$

$$\begin{aligned}\Delta(b) &= \left[P_1 \cdot D_k + P_3 \cdot (C_b - D_k)\right] - \left[P_1 \cdot D_j + P_3 \cdot (C_b - D_j)\right] \\ &= P_1 \cdot (D_k - D_j) + P_3 \cdot (D_j - D_k)\end{aligned}$$

In this case Job c is tardy in I_j and will be tardy if assigned to I_k.

$$\Delta(c) = \text{Cost of C in } I_k - \text{Cost of C in } I_j$$

$$= \left[P_1 \cdot D_k + P_3 \cdot (C_c - D_k)\right] - \left[P_1 \cdot D_j + P_3 \cdot (C_c - D_j)\right]$$

$$= \Delta b$$

Thus, Job b can be assigned to I_j and Job c can be assigned to I_k without increasing the total cost.

Case 3: Job b tardy in I_k but early in I_j, or $D_k \le C_b \le D_j$

$$\Delta(b) = \left[P_1 \cdot D_k + P_3 \cdot (C_b - D_k)\right] - \left[P_1 \cdot D_j + P_2 \cdot (D_j - C_b)\right]$$

$$= P_1 \cdot (D_k - D_j) + C_b \cdot (P_3 + P_2) - (P_3 \cdot D_k + P_2 \cdot D_j)$$

In this case, Job a is early in I_j, and early or tardy in I_k.

$$\Delta(a) = \begin{cases} P_1 \cdot (D_k - D_j) + P_2 \cdot (D_k - D_j) & \text{if } C_a \le D_k \\ P_1 \cdot (D_k - D_j) + C_a \cdot (P_3 + P_2) - (P_3 \cdot D_k + P_2 \cdot D_j) & \text{if } C_a \ge D_k \end{cases}$$

It is easy to see that $\Delta(a) \le \Delta(b)$. Thus, Job a can be assigned to I_k and Job b can be assigned to I_j without increasing the total cost.

Case 4: Job b early in I_k but tardy in I_j, or $D_j \le C_b \le D_k$

$$\Delta(b) = \left[P_1 \cdot D_k + P_2 \cdot (D_k - C_b)\right] - \left[P_1 \cdot D_j + P_3 \cdot (C_b - D_j)\right]$$

$$= P_1 \cdot (D_k - D_j) - C_b \cdot (P_2 + P_3) + (P_2 \cdot D_k + P_3 \cdot D_j)$$

In this case, Job c is tardy in I_j and early or tardy in I_k.

$$\Delta(c) = \begin{cases} (P_1 - P_3) \cdot (D_k - D_j) & \text{if } C_c \ge D_k \\ P_1 \cdot (D_k - D_j) - C_c \cdot (P_2 + P_3) + (P_2 \cdot D_k + P_3 \cdot D_j) & \text{if } C_c \le D_k \end{cases}$$

Since $\Delta(c) \le \Delta(b)$, Job c can be assigned to I_k and Job b can be assigned to I_j without increasing the total cost.

This completes the proof.

Lemma 2: The TP(D,I,σ) problem has an optimal schedule with zero machine-idle time.

Proof. Consider a schedule in which the machine is idle in the interval $(t,t+\delta)$ and Job ℓ starts at $t+\delta$. We show that an alternate feasible schedule can be constructed such that there is zero idle time in the interval $(t,t+\delta)$ and the total cost does not go up.

We complete the proof by considering the following two cases.

Case 1: Job ℓ is tardy; i.e., $D_\ell < t+\delta+t_\ell$.

Revise the schedule by moving Job ℓ backward by $\varepsilon = \min(\delta, t+\delta+t_\ell-D_\ell)$. Job ℓ starts at $t+\delta-\varepsilon$ in the revised schedule. Either the machine-idle time in the interval $(t,t+\delta)$ goes to zero or Job ℓ completes on time (or both) in the revised schedule. The cost of the revised schedule is lower because the tardiness of Job ℓ has gone down.

Case 2. Job ℓ is early or on-time; i.e., $D_\ell \geq t+\delta+t_\ell$.

Let Job ℓ be in position n. Revise the schedule by moving the jobs in positions n, n+1,...,N backward by δ. Also, reduce the due dates of all these jobs by δ. The revised solution costs less because it has lower earliness and due date penalties.

This completes the proof.

Lemma 3: Let D, I, and σ be known. Let D be such that there is a D_j that does not coincide with the completion time of any job in I_j then it is possible to find an alternate D_j without increasing the total cost.

Proof: Let T_j be the number of tardy jobs and E_j be the number of early jobs in I_j. Let a and b be two adjacent jobs in σ such that $C_a < D_j < C_b$. Δ is a small quantity such that $D_j-\Delta > C_a$ and $D_j+\Delta C_b$. Let Δ^+ denote the increase in the total cost for jobs in I_j if D_j is increased by Δ and let Δ^- denote the increase in the total cost if D_j is reduced by Δ. We have

$$\Delta^+ = n_j \cdot \Delta \cdot P_1 + E_j \cdot \Delta \cdot P_2 - T_j \cdot \Delta \cdot P_3, \text{ and}$$

$$\Delta^- = -n_j \cdot \Delta \cdot P_1 - E_j \cdot \Delta \cdot P_2 + T_j \cdot \Delta \cdot P_3 = -\Delta^+.$$

If $\Delta^+ \geq 0$, we can decrease D_j upto C_a without increasing the total cost. If $\Delta^- \geq 0$, we can increase D_j upto C_b without increasing the cost. This completes the proof.

Proof of Lemma 5:

We prove the lemma for i = 1 and 2. The proof can be easily extended for i = 3, 4, ..., N.

Proof for $WT_1(n) \geq WT_1\left(\left(\frac{N}{2}\right)^+\right)$:

Consider the problem with m = 2 and $t_1 = t_2 = \ldots = t_N = t$. (Since $WT_j(n)$'s are independent of processing times, it is sufficient to prove this result for $t_1 = t_2 = \ldots = t_N = t$.) From (8), we get

$$WT_1(n) = \frac{COST(n)}{t}, \text{ and } WT_1\left(\left(\frac{N}{2}\right)^+\right) = \frac{COST\left(\left(\frac{N}{2}\right)^+\right)}{t}.$$

To prove $WT_1(n) \geq WT_1\left(\left(\frac{N}{2}\right)^+\right)$, it is sufficient to show that

$$COST(n) \geq COST\left(\left(\frac{N}{2}\right)^+\right) \qquad \text{(A1)}$$

for $1 \leq n \leq \left(\frac{N}{2}\right)^+$. Note that $COST(\ell)$ here denotes the minimum total cost for the N-job problem with m = 2 due dates, $t_1 = t_2 = \ldots = t_N$ and ℓ jobs assigned to D_1. We provide a proof by construction.

Consider a COST(n)-solution with n < N-n. Let $E_1'(T_1')$ denote the number of early (tardy) jobs in Batch 1 and $E_2'(T_2')$ denote the number of early (tardy) jobs in Batch 2 in the COST(n)-solution. It is easy to see that $n = E_1'+T_1'$, $D_1 = E_1'\cdot t$, $D_2 = (E_1'+E_2'+T_1')\cdot t$,

Due date penalty for the first n jobs $= n\cdot D_1\cdot P_1$

$= (E_1'+T_1')\cdot(E_1'\cdot t)\cdot P_1$,

Earliness/Tardiness Cost for the first n jobs $= P_2\cdot((E_1'-1)\cdot t+(E_1'-2)\cdot t +\ldots+1\cdot t) +$

$P_3(t+2t+\ldots+T_1'\cdot t)$

$$= \frac{P_2 \cdot E_1' \cdot (E_1'-1) \cdot t}{2} + \frac{P_3 \cdot T_1' \cdot (T_1'+1) \cdot t}{2}$$

Due date penalty for the last N-n jobs $= (N-n) \cdot D_2 \cdot P_1$,

$$= (E_2'+T_2') \cdot (E_1'+E_2'+T_1') \cdot t \cdot P_1,$$

Earliness/Tardiness Cost for the last N-n jobs $= \frac{P_2 \cdot E_2' \cdot (E_2'-1) \cdot t}{2} + \frac{P_3 \cdot T_2' \cdot (T_2'+1) \cdot t}{2}$, and

$$\text{COST}(n) = (E_1'+T_1') \cdot (E_1' \cdot t) \cdot P_1 + \frac{E_1' \cdot (E_1'-1) \cdot t \cdot P_2}{2} + \frac{T_1' \cdot (T_1'+1) \cdot t \cdot P_3}{2} +$$

$$(E_2'+T_2') \cdot (E_1'+T_1'+E_2') \cdot t \cdot P_1 + \frac{E_2' \cdot (E_2'-1) \cdot t \cdot P_2}{2} + \frac{T_2' \cdot (T_2'+1) \cdot t \cdot P_3}{2}.$$

Now consider another feasible solution constructed from the COST(n)-solution by changing the assignment for the first job in Batch 2. We assume that the assignment for the first job in Batch 2 is changed to Batch 1. Thus the first (n+1) jobs are now assigned to Batch 1 and the last N-(n+1) jobs are assigned to Batch 2. The due dates D_1 and D_2 are determined as follows.

CASE 1: $D_1 = (E_1'+1) \cdot t$ and $D_2 = (E_1'+E_2'+T_1') \cdot t$

Let COST_1 denote the cost for this case, we have

$$\text{COST}_1 = (E_1'+T_1'+1) \cdot (E_1'+1) \cdot t \cdot P_1 + \frac{(E_1'+1) \cdot E_1' \cdot t \cdot P_2}{2} + \frac{T_1' \cdot (T_1'+1) \cdot t \cdot P_3}{2} +$$

$$(E_2'+T_2'-1) \cdot (E_1'+T_1'+E_2') \cdot t \cdot P_1 + \frac{(E_2'-1) \cdot (E_2'-2) \cdot t \cdot P_2}{2} + \frac{T_2' \cdot (T_2'+1) \cdot t \cdot P_3}{2}, \text{ and}$$

$$\text{COST}(n) - \text{COST}_1 = t \cdot (P_1+P_2) \cdot (E_2'-E_1'-1) \qquad \text{(A2)}$$

CASE 2: $D_1 = E_1' \cdot t$ and $D_2 = (E_1'+T_1'+E_2'+1) \cdot t$

Let COST_2 denote the cost for this case, we have

$$\text{COST}_2 = (E_1'+T_1'+1) \cdot E_1' \cdot t \cdot P_1 + \frac{E_1' \cdot (E_1'-1) \cdot t \cdot P_2}{2} + \frac{(T_1'+1) \cdot (T_1'+2) \cdot t \cdot P_3}{2} +$$

$$(E_2'+T_2'-1)\cdot(E_1'+T_1'+E_2'+1)\cdot t\cdot P_1 + \frac{E_2'\cdot(E_2'-1)\cdot t\cdot P_2}{2} + \frac{(T_2'-1)\cdot T_2'\cdot t\cdot P_3}{2}, \text{ and}$$

$$COST(n) - COST_2 = (T_2'-T_1'-1)\cdot(P_3-P_1)\cdot t \qquad (A3)$$

Since $n < N-n$, we have either $E_1' < E_2'$ and $T_1' \le T_2'$, or $E_1' \le E_2'$ and $T_1' < T_2'$. If $E_1' < E_2'$, then $COST(n) \ge COST_1$, and if $T_1' < T_2'$, then $COST(n) \ge COST\ 2$. Thus, for $n < N-n$, an alternate feasible solution with an equal or lower cost can be constructed by reassigning a job from Batch 2 to Batch 1. This implies $WT_1(n) \ge WT_1(n+1)$ for $n < N-n$. This completes the proof for $WT_1(n) \ge WT_1\left(\left(\frac{N}{2}\right)^+\right)$.

Proof for $WT_2(n) \ge WT_2\left(\left(\frac{N}{2}\right)^+\right)$:

Consider the problem with $m = 2$, $t_1 = 0$ and $t_2 = t_3 \ldots = t_N = t$. From (8), we get

$$WT_2(\ell) = \frac{COST(\ell)}{t} \text{ for } \ell \in \langle 1, N-1\rangle.$$

To prove $WT_2(n) \ge WT_2\left(\left(\frac{N}{2}\right)^+\right)$ for $n \in \langle 1, \left(\frac{N}{2}\right)^+\rangle$, it is sufficient to prove that

$$COST(n) \ge COST\left(\left(\frac{N}{2}\right)^+\right) \text{ for } n \in \langle 1, \left(\frac{N}{2}\right)^+\rangle.$$

Again, a proof by construction is provided. Consider a COST(n)-solution with E_1', T_1', E_2' and T_2' as defined above. The job with zero processing time is counted in E_1'. We have $D_1 = (E_1'-1)\cdot t$ and $D_2 = (E_1'+T_1'+E_2'-1)\cdot t$. The zero processing time job completes at D_1. We get

$$COST(n) = (E_1'+T_1')\cdot(E_1'-1)\cdot t\cdot P_1 + \frac{(E_1'-1)\cdot(E_1'-2)\cdot t\cdot P_2}{2} + \frac{(T_1'+1)\cdot T_1'\cdot t\cdot P_3}{2} +$$

$$(E_2'+T_2')\cdot P_1\cdot(E_1'+T_1'+E_2'-1)\cdot t + \frac{E_2'\cdot(E_2'-1)\cdot t\cdot P_2}{2} + \frac{T_2'\cdot(T_2'+1)\cdot t\cdot P_3}{2}.$$

Now consider an alternate feasible solution such that Batch 1 has the first n+1 jobs and Batch 2 has the last N-(n+1) jobs in the sequence for the COST(n)-solution. The zero processing time job completes at D_1 where D_1 and D_2 are determined as follows:

CASE 1: $D_1 = E_1' \cdot t$ and $D_2 = (E_1' + T_1' + E_2' - 1) \cdot t$. We get

$$COST_1 = (E_1' + T_1' + 1) \cdot (E_1' \cdot t) \cdot P_1 + \frac{E_1' \cdot (E_1' - 1) \cdot t \cdot P_2}{2} + \frac{T_1' \cdot (T_1' + 1) \cdot t \cdot P_3}{2} +$$

$$(E_2' + T_2' - 1) \cdot (E_1' + T_1' + E_2' - 1) \cdot t \cdot P_1 + \frac{(E_2' - 1) \cdot (E_2' - 2) \cdot t \cdot P_2}{2} + \frac{T_2' \cdot (T_2' + 1) \cdot t \cdot P_3}{2}, \text{ and}$$

$$COST(n) - COST_1 = (E_2' - E_1' - 1) \cdot t \cdot P_1 \qquad (A4)$$

CASE 2: $D_1 = (E_1' - 1) \cdot t$ and $D_2 = (E_1' + T_1' + E_2') \cdot t$. We get

$$COST_2 = (E_1' + T_1' + 1) \cdot (E_1' - 1) \cdot t \cdot P_1 + \frac{(E_1' - 1) \cdot (E_1' - 2) \cdot t \cdot P_2}{2} + \frac{(T_1' + 1)(T_1' + 2) \cdot t \cdot P_3}{2} +$$

$$(E_2' + T_2' - 1) \cdot (E_1' + T_1' + E_2') \cdot t \cdot P_1 + \frac{E_2' \cdot (E_2' - 1) \cdot t \cdot P_2}{2} + \frac{(T_2' - 1) \cdot T_2' \cdot t \cdot P_3}{2}, \text{ and}$$

$$COST(n) - COST_2 = (T_2' - T_1' - 1) \cdot (P_3 - P_1) \cdot t \qquad (A5)$$

Note that (A4) is the same as (A2) and (A5) is the same as (A3). As in the $WT_1(\cdot)$ case, we can argue $WT_2(n) \geq WT_2(n+1)$ for $n < N-n$. This completes the proof for $WT_2(\cdot)$.

The above proof can be extended to show $WT_i(n) \geq WT_i(n+1)$ for $n < N-n$ and $i \in \langle 3,N \rangle$. To prove it for $i = k$, it will be assumed that the 2-due date N-job problem has (k-1) jobs with zero processing times, and N-(k-1) jobs with processing time of $t > 0$ each. All the zero processing time jobs are assumed to complete at D_1. For a given COST(n) solution, an alternate feasible solution is constructed by increasing the number of jobs assigned to Batch 1 from n to n+1. Two alternatives for due dates are considered. Compared to the due dates in COST(n), the due date for Batch 1 is increased by t while the due date for Batch 2 is kept the same in Case 1. In Case 2, the due date for Batch 1 is kept the same while the due date for Batch 2 is increased by t.

References

1. Bagchi, U., "Due Date or Deadline Assignment to Multi-Job Orders to Minimize Total Penalty in the One Machine Scheduling Problem," Working Paper, University of Texas at Austin, 1989.

2. Baker, K.R. and J. Bertrand, "A Comparison of Due-Date Selection Rules," AIIE Transactions, 1981, 13, 123-131.

3. Baker, K.R. and G.D. Scudder, "Sequencing with Earliness and Tardiness Penalties: A Review," Operations Research, 1990, 38, 22-36.

4. Bector, C.R., Y.P. Gupta and M.C. Gupta, "Determination of an Optimal Common Due Date and Optimal Sequence in a Single Machine Job Shop," International Journal of Production Research, 1988, 26, 613-628.

5. Panwalkar, S.S., M.L. Smith and A. Seidmann, "Common Due Date Assignment to Minimize Total Penalty for the One Machine Scheduling Problem," Operations Research, 1982, 30, 391-399.

6. Eilon, S. and I.J. Chowdhary, "Due Date in Job Shop Scheduling," International Journal of Production Research, 1976, 14, 223-237.

7. Seidman, A., S.S. Panwalkar and M.L. Smith, "Optimal Assignment of Due Dates for a Singh Processor Scheduling Problem," International Journal of Production Research, 1981, 19, 393-399.

8. Shanthikumar, J.G. and D.D. Yao, "On Server Allocation in Multiple Center Manufacturing Systems," Operations Research, 1988, 36, 333-342.

9. Weeks, J.K., "A Simulation Study of Predictable Due Dates," Management Science, 1979, 25, 363-373.

10. Weeks, J.K. and J.S. Fryer, "A Methodology for Assigning Minimum Cost Due Dates," Management Science, 1977, 23, 872-881.